1 Lab+Hwk 3: Introduction to the miniature Khepera robot

This laboratory requires the following equipment:

- C programming tools (gcc, make).
- Webots simulation software.
- Webots User Guide and Reference Manual.
- A full Khepera kit: a RS232 serial cable with integrated adapter DB9 to DB25, a serial RS232 interface with dedicated robot serial cable, a cable unroller, a 9V continuous power supply unit, and a Khepera robot (basic module).
- Khepera User Manual.

The laboratory duration is about 2 hours. Homework will be due on **the 6th day after your lab session, at 5 p.m**. Please copy your solutions, plots, descriptions (*.doc, *.ps, *.html, *.pdf, *.txt), movies, and so on into the directory *~ correll/lab03/yourusername/*. The directory *~ correll/lab03* allows you to write but not to read. The corresponding Webots files (see section 2 for details) should be saved in a sub- directory *webots/*. Please strictly follow these guidelines to facilitate the grading process, and keep in mind that no late solutions will be accepted unless motivated by health reasons (doctor certificate required).

1.1 Grading

In the following text you will find several exercises and questions. The notation $\mathbf{I_x}$ means that the problem has to be solved by implementing a piece of code and performing a simulation. The notation $\mathbf{S_x}$ means that the question can be solved using only additional simulation. The notation $\mathbf{Q_x}$ means that the question can be answered theoretically without any simulation. **Please use this notation in your answer file**!

The number of points for each exercise is given between parentheses. **Sometimes there will be an asterisk (*) instead of a number of points; this indicates that no code or answer is produced in this part, and the points will instead be awarded based on how well you answer the question(s) related to the problem in the homework section.** The total sum of the points for the laboratory and homework part is 100.

1.2 Robot's overview

Khepera is a miniature mobile robot developed to perform "desktop" experiments. Its distinguishing characteristic is its small size (5.5 cm in diameter). Other basic features are: significant processing power (Motorola® 68331, 32 bit processor running at 16 MHz), a 128 kB EEPROM (which can be extended to 512 kB) and 256 kB of SRAM, and an energetic autonomy of about 45 minutes (with no additional turrets) through the onboard batteries.

As we have seen in the Webots labs, Khepera is equipped with two motors and a belt of eight proximity sensors. Every wheel is controlled by a DC motor through a 25:1 reduction gear box. An incremental, magnetic encoder is placed on each motor

axis and returns 24 pulses per revolution of the motor. In turn, this allows a resolution of 600 pulses per revolution of the wheel. Considering the diameter of the wheel, this corresponds to 12 pulses per millimeter of path of the robot or 0.08 mm per pulse (see page 6 of the Khepera User Manual for more details and description of command "H" of the serial protocol at page 36).

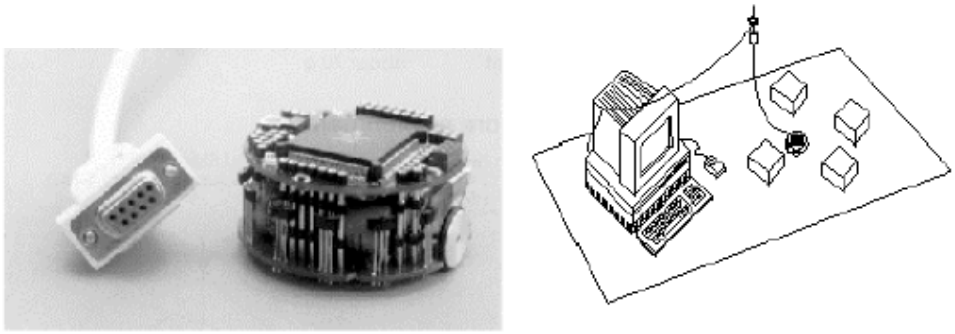
The proximity sensors consist of an integrated IR emitter and receiver (SFH900-2 of Siemens®). These integrated devices allow two types of measurements:

- Measurement of the ambient light. This measurement is performed by using only the receiver, without any emitting activity. The sample rate of each sensor is 20 ms; activation and reading of a single sensor takes 2.5 ms and it is performed in a sequential way. Notice that basically all the photosensitive components currently available on the market have the sensitivity peak around the near infrared range. We therefore mainly measure the IR component of the ambient light.
- Measurement of the light reflected by obstacles. This measurement is performed by using also the integrated emitter. The returned value represents the difference between two measurements, one with active light emission and one without (ambient light). The sample rate of each sensor is 20 ms; activation and reading of a single sensor takes 2.5 ms and it is performed in a sequential way.

The output of each type of measurement is an analog value which is then converted in a 10-bit digital value by a dedicated A/D converter. More details are given at page 8 to 11 of your Khepera user manual.

1.3 Serial link

As we have seen in the Webots lab, Khepera can be controlled in three different ways: via a serial link protocol, by downloading the control program to the SRAM of the Khepera, or by creating a customized EEPROM.



a b

Figure 1-1: a) Close-up of a Khepera robot with the corresponding RS232 connector. b) Overview of the set-up used in this lab.

In this lab, we will control exclusively the robot via the serial link, which in turn means that your (high-level) control algorithm will run on the workstation and Khepera will execute the commands required by your perception-to-action loop (see *Figure 1-1*). In this case, at the Khepera level are only performed all low-level control routines such as speed regulation of motors or sequential scanning of the sensors. A part of the BIOS of Khepera is programmed to manage this type of tasks.

Remark: Khepera can be controlled through the serial line at three different speeds: 9600, 19200, and 38400 bauds, according to the chosen run mode and jumper configuration (see Appendix E, page 48 of the user manual).

1.4 Programming via the serial link

In this lab, certain information must be transferred between robots. This could be accomplished using wireless radio transmissions or by broadcasting data with infrared. Looking closer at the three *Khepera* robots in *flocking1.wbt*, one can see two additional nodes, namely an emitter and a receiver node, allowing for emitting and receiving data over a fixed range and a dedicated, arbitrary channel[1]. This will be used to allow robots to track wheel speeds (part 2.2, 3) and relative positions (part 4) of other robots.

2 Lab: Getting familiar with Khepera in serial control mode

This lab consists of several modules. We will start by understanding what can be achieved with the Khepera serial protocol. We will then move to an exercise for evaluating the odometric performances of Khepera. Finally, we will implement some simple Braitenberg vehicles on the real Khepera, taking advantage of the expertise accumulated with the Webots simulator. We will in particular compare the performances of the **same** control algorithm, run on the same robot, through Webots, through a direct control of the serial line with C code, and with the control program completely running on the robot (demo integrated in the BIOS).

2.1 Getting started with Khepera

First at all, copy all the files needed for the lab in your home directory:

```
cp -r ~pugh/khepera .
```

Check your hardware set-up:

- Set the jumpers of your robot in **Mode 3** (see pages 9-10 of the Khepera2 User Manual) for a serial connection at 38400 baud.
- **First** connect the robot to the dedicated serial cable, and **then** plug in the power supply of your serial interface (pages 17-18 of the Khepera User Manual). Invert the sequence (unplug the power supply first and then disconnect the robot) when you are leaving. When you plug in the power, you should notice a blink of the green led on the serial interface indicating that everything is all right. A red led switched on continuously signals a problem in the serial connection. Try to reset the robot (reset button) or call the teaching assistant.
- Check that the battery switch (page 8 of Khepera2 User Manual) is in **off** position. Power supply is provided by the serial cable; if the battery switch is on and your battery is discharged, then the robot will not be able to exploit the cable external energy source and to respond to your commands.

Launch the minicom serial interface by entering the `minicom ka38` (= Khepera configuration, port A, 38400 Bauds) command on your terminal. Press the reset button on the robot; your terminal should display:

```
ROM of minirobot KHEPERA,...
```

With *ctrl-A Z* you obtain the main menu of minicom. Configure your terminal so that you can have a local echo (type *E* and *return*). In order to exit from all pop-up menu in minicom, work with *esc* button (unmodified configuration) or with return (updated configuration). Exit from minicom is performed by typing with *ctrl-A X*. **Do not exit by simply closing the terminal window in which you are working**: this will lock your serial port and you need the help of the system administrator (super-user rights required) to unlock it!

Enter `list` in your minicom terminal. Your Khepera will answer with all the commands it understands through the serial line. The most important for this lab are the protocol orders which are described from page 35 to 39 of the user manual.

There is also an on-line help for each command: type `help D` to obtain information about setting motor speeds. Test some commands to control actuators (speed, position) and reading sensors (proximity, light, and wheel encoders).

2.2 Mobility and odometry on Khepera

Q_1 (10): Fig 2-1 shows a Khepera performing a rotation of an established radius and arc length. How can you perform such a movement with the minimal number of commands? Keep in mind that the default is for both robot motors to turn at the same rate.

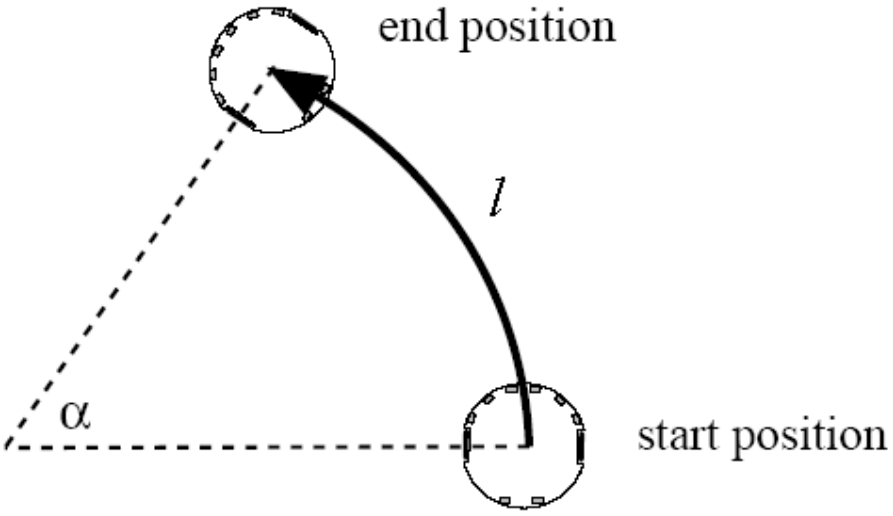


Figure 2-1: Rotation of Khepera of angle α and arc length *l*.

S_2 (*): Perform the movement that corresponds to going 2000 units on the right wheel and 1000 units on the left wheel and measure the robot displacement. Hint: marking the position of the Khepera can be tricky. You may want to mark (with a **pencil**, not pen) the position of each wheel and then find the

point that bisects the line between them. For measuring, use the rulers provided by the teaching assistants.

2.3 Controlling the robot via the serial line with a simple algorithm

In your *khepera/* directory, you will find a simple Khepera algorithm called *kalgo1.c*.

S_3(3): Exit from the minicom terminal. Compile the program *kalgo1* by typing "make kalgo1", and run it. Why is the Khepera continuously rotating even if the program execution has finished?

I_4(3): Modify this program so that the Khepera stops at the end of the program. Save the program under kalgo2.c.

I_5(10): For more complicated programs, the readability of the code gets worse if every time that we have to send a command to Khepera, we have to explicitly use in the main program protocol strings on the serial port functions. Modify kalgo2.c by introducing a dedicated function which sends a speed command for both wheels on the serial port. Implement the function with the following name and parameters: khepera_set_speed (serialport, speed), where serialport specifies the descriptor of the serial port and speed specifies a vector for the motor speeds (left, right, range: -20, +20). Save your program under kalgo3.c.

2.4 Implementing Braitenberg vehicles

Copy the *braiten.wbt* world and *braiten* controller to your Webots directory.

In your *khepera/* directory, you will find a simple Khepera algorithm called *kbraiten1.c* which exploits functions for sensor reading and motor controlling similar to those used in Webots. We have performed this translation for you (imported via the *web2kheS.h* header file).

I_6(10): Taking advantage of your expertise accumulated in the Webots lab, implement a Braitenberg vehicle for obstacle avoidance. Hint: translate the braiten.c file contained in the directory ~pugh/webots/controllers/braiten/ into a new file kbraiten2.c for serial control of the real robot. Use the functions defined in web2khe.c, h.

We want now compare three different running modes of the same controller: through Webots, via the serial line, and with the demo version implemented on the robot.

R_7(*): Make sure that you have closed the minicom terminal and that your are not trying to control the serial port from one of your programs. Launch Webots and load the *braiten.wbt* world. You should have automatically loaded the Braitenberg controller. Double click on the robot. Set the jumper configuration on the simulated and on the real robot on 19200 bauds. Switch from simulated to serial control of the robot (page 38 of the Webots User Guide), and press ‘fast forward’. Note the performance of the controller.

Hint: use the wood frame provided on your desk as arena for your tests. Change the baudrate to 38400 and test again. Note the new performance.

R_8(*): Use the black floor for your arena and again perform your tests at 19200 and 38400 baud. Note the performance.

R_9(*): Exit from Webots, and run your kbraiten2.c algorithm at 19200 and at 38400 bauds on the white floor. Note the performance.

R_{10}(*): Run the demo in the Khepera ROM. In this case the control algorithm is completely running on the on-board microcontroller. Run the demo in two different ways. First, from minicom, type "run demo" and return. Second, use the real robot with the correct jumper configuration (mode 0, page 48 of the User Manual). Remember to enable each new jumper configuration with the reset button. Note the performances. If you want, take away the cable, switch on the battery, and the robot should run start up running the Braitenberg demo if the jumper configuration is mode 0.

3 Homework

Q_{11} (8): What is the angle α [degree] and the arc length l defined by the robot center [mm] of a forward movement performed with 2000 units on the right wheel and 1000 on the left one?

Q_{12} (5): Compare your calculated displacement and measured displacement of the Khepera in **S_2**. What is the experimental error generated by the inaccuracy in the odometry?

Q_{13} (10): What would cause inaccuracy in odometry? Would this inaccuracy be larger when the robot was going straight or when it was turning? Why?

Q_{14} (8): How could you produce the same movement on the Khepera using only the "D" and "H" commands? What are the limitations and constraints of this approach?

Q_{15} (10): Recall the performances of the Webot-controlled Khepera in **R_7**. How did the controller perform in the real world? How did the performance change with the baud rate? Why?

Q_{16} (5): Recall the performance of the Webot-controlled Khepera in **R_8** with the black floor. How is the performance in avoiding the walls in comparison to the white floor? Why? Hint: consider the mechanism of an IR proximity sensor studied in the previous labs.

Q_{17} (3): Recall the performance of the kbraiten2.c algorithm in **R_9**. Is there a difference in the performance from the Webot-controlled Khepera algorithms?

Q_{18} (5): Recall the performance of the demo in the Khepera ROM in **R_{10}**. How does its performance in obstacle avoidance compare to the other methods? Why?

Q_{19} (10): Odometry in Multi-Robot Systems: We have been using a single robot doing obstacle avoidance of the walls. We can use the same algorithm with multiple robots in the arena, where they avoid each other as well as the walls. How do you think this change would affect the accuracy of the odometry of the Khepera robots over time? Why? Give several reasons.